(U) High Scalability Video ISR Exploitation

October 2012

Michael Czajkowski and Daniel Donavanik

Lockheed Martin Advanced Technology Laboratories
3 Executive Campus, Cherry Hill NJ 08002
{michael.czajkowski, daniel.donavanik}@lmco.com

(U) **Keywords:** Computer vision, big data, cloud computing, Hadoop, Map/Reduce, scene understanding, visual saliency, scalability, ISR, and Motion Intelligence

(U) ABSTRACT

(U) The Intelligence Community uses computer vision (CV) algorithms on data from state-of-the-art sensors and platforms (e.g. Autonomous Real-Time Ground Ubiquitous Surveillance, ARGUS) on the National Image Interpretability Rating Scale (NIIRS) at level 6. Ultra-high quality cameras like the Digital Cinema 4K (DC-4K), which recognizes objects smaller than people, will be available soon for Wide-Area Motion Intelligence (WAMI). However, even if a platform was equipped with DC-4K, it would useless without CV algorithms to quickly process the vast quantities of data captured.

(U) Today, several CV algorithms scale up by partitioning data across multiple nodes of computation. The standard approach is to increase the amount of processing power (e.g. GPUs) available. However, to achieve NIIRS level 7+ an emerging problem must be addressed: hard disk read latency. Reductions in disk read latency have not kept pace with increases in volume from 1990 through today. For example, reading a 3TB full disk from beginning to end takes hours. Considering that a NIIRS-8 DC-4K camera captures 32MB of data per frame, it would capture 3.2TB of data in a single mission hour. Industry has already solved this "big data" problem in large-scale text processing through cloud computing architectures like Apache Hadoop. Hadoop applies a parallel batch-processing paradigm that reads data from multiple hard disks simultaneously called Map/Reduce. In contrast to Hadoop, Modern CV algorithms assume a sequential data stream being read in order. While certain computations can be made in parallel already, integrating and correlating features computed simultaneously from randomly accessed portions of the data stream is an unmet challenge.

(U) We present High Scalability Video ISR Exploitation (HVISRE), a novel approach to leveraging the Map/Reduce cloud computing framework for targeted pattern recognition and threat detection in large video data corpuses. HVISRE's pattern recognition is based on neuromimetic attention, a technique for bottom-up computation of visual saliency inspired by the mammalian visual cortex. The algorithm integrates localized center-surround features to rapidly compute a "saliency map" indicating regions of imagery that are likely to contain patterns of interest. The integration scheme is

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
OCT 2012	**N/A**	**-**

4. TITLE AND SUBTITLE
High Scalability Video ISR Exploitation

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Martin Advanced Technology Laboratories 3 Executive Campus, Cherry Hill NJ 08002

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM202976. 2012 Joint Meeting of the Military Sensing Symposia (MSS) held in Washington, DC on October 22-25, 2012.

14. ABSTRACT
(U) The Intelligence Community uses computer vision (CV) algorithms on data from state-of-the-art sensors and platforms (e.g. Autonomous Real-Time Ground Ubiquitous Surveillance, ARGUS) on the National Image Interpretability Rating Scale (NIIRS) at level 6. Ultra-high quality cameras like the Digital Cinema 4K (DC-4K), which recognizes objects smaller than people, will be available soon for Wide-Area Motion Intelligence (WAMI). However, even if a platform was equipped with DC-4K, it would useless without CV algorithms to quickly process the vast quantities of data captured. (U) Today, several CV algorithms scale up by partitioning data across multiple nodes of computation. The standard approach is to increase the amount of processing power (e.g. GPUs) available. However, to achieve NIIRS level 7+ an emerging problem must be addressed: hard disk read latency. Reductions in disk read latency have not kept pace with increases in volume from 1990 through today. For example, reading a 3TB full disk from beginning to end takes hours. Considering that a NIIRS-8 DC-4K camera captures 32MB of data per frame, it would capture 3.2TB of data in a single mission hour. Industry has already solved this "big data" problem in large-scale text processing through cloud computing architectures like Apache Hadoop. Hadoop applies a parallel batchprocessing paradigm that reads data from multiple hard disks simultaneously called Map/Reduce. In contrast to Hadoop, Modern CV algorithms assume a sequential data stream being read in order. While certain computations can be made in parallel already, integrating and correlating features computed simultaneously from randomly accessed portions of the data stream is an unmet challenge.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**15**	

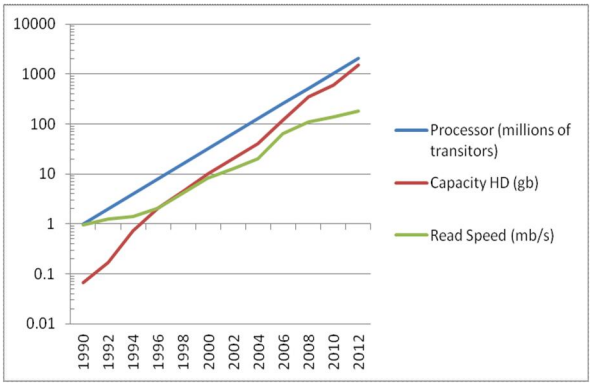
Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

tunable for a variety of mission scenarios and imagery domains, as what is "salient" may differ between geographic environments and conditions under which the imagery was obtained. Our novel approach to the big data partitioning problem exploits the modular structure of the saliency algorithm itself; low-level computation of conspicuity maps from individual center-surround features not relying upon global situation information. We have developed a technique for deploying these fundamental units of work as "mapper" jobs in order to effectively exploit Hadoop's mature infrastructure. The goal of our approach is to assist the analyst in quickly determining which regions of an image (or frames in a WAMI video) merit in-depth scrutiny in a data domain where the vast majority of captured images are likely to contain little information of value.

1.0 (U) Introduction

(U) The Intelligence Community uses Computer Vision (CV) algorithms to process imagery from state-of-the art platforms (e.g. Shadow Harvest, Blue-Devil, ARGUS, etc.) to build information for actionable intelligence. These sensors capture images at National Image Interpretability Rating Scale (NIIRS) level 6 where the usual detectable image is the size of a truck [14]. These sensors are getting cheaper each year, and soon it will be viable to purchase ultra-high quality cameras like the Digital Cinema 4K (DC-4K) for use in the field. However, even if such a UAV sensor with a DC-4K was flown, it would be fruitless if CV algorithms could not process the *immense* quantities of data they capture in a *reasonable* amount of time.

(U) However in big data processing, reductions in read latencies have not kept up pace with increases in volume and CPU processor transistor count since 1990. See Figure 1, which is based on purchasing an average processor and hard-disk. In that time, both capacity and transistor count have increased exponentially while hard drive read speeds have increased closer to linearly. Table 1 shows that a video that would occupy a full hard disk today would take much longer to process than a video that occupied a full hard disk years 20 years ago.



(Unclassified)

Figure 1. (U) Hard Drive Read Speed, Capacity and CPU Processing Over Time

(U) Considering that to do NIIRS 8, a DC-4K camera used captures 32MB of data per frame and thus generates 3.2TB of data in a single hour of a mission. As Table 1 shows, a lot of time will be spent processing that data sequentially from one disk. Therefore a parallel processing technique must be adapted by the CV world to allow multiple disks to be read and analyzed simultaneously.

(U) The typical industry solution is one of two options: scaling-up or scaling-out. Those who follow the scaling-up approach increase processing by buying better and more expensive computing hardware. This approach is limited by availability and cost. For example, consider the Oracle Exadata System that stores 100TB, processes bandwidth at 25 GB/s and costs $1.6 million [17]. This system is considered the

Table 1. (U) Capacity vs. Read Speed Over 20 Years of IT Evolution

Year	Capacity	Read Speed	Full Read Time
1990	1.33GB	4.4 MB/s	5.1 minutes
2010	2TB	138 MB/s	4.2 hours

(Unclassified)

(U) top-of-the line as the world's fastest database machine; no single product out there could process 100TB data faster on its own. Once this capacity limit is reached and the cost paid, the only solution left is to introduce parallel processing to simultaneously read multiple disks from multiple machines. This is called a "scaled out" solution when large quantities of commodity drives and CPUs are used. Scaled-out solutions don't have the same upper limit that a scaled-up solution does. Simply buying another set of computers increases their capacity to process linearly until power supply requirements can no longer be met [23].

(U) Looking at today, the CV community has largely been following a multi-processor solution by applying more processing resources with GPUs to solve this particular problem. However, only a few people [12, 16] have considered how CV algorithms might read data off of multiple disks in parallel. The difficulty lies in reading parts of a video stream from different hard disks simultaneously and reaching the same conclusions that a single-node solution reaches.

(U) Inspired by big-data cloud computing approaches, Lockheed Martin Advanced Technology Laboratories (ATL) has been researching and building CV algorithms that use multiple processors and read video from multiple disks simultaneously. We present our research and introduce an adaptation of a pattern recognition CV algorithm called "Visual Saliency" that uses Apache Hadoop, a popular cloud computing architecture designed for scaled-out big-data analysis [2]. The paper is organized as follows: Section 2 gives a background of Apache Hadoop and the state of the art in computer vision. We describe what characteristics of computer vision feasible in Hadoop through presenting our research. Section 3 introduces an implementation of Visual Saliency in Hadoop as a target-finding algorithm that can detect and describe target imagery in sparse Wide Area Motion Intelligence (WAMI) video data sets. Section 4 describes experimental results where we used Visual Saliency to find a truck object in GISR video data. We tested the implementation of our Visual Saliency algorithm against two metrics: quality of results and speed compared to non-cloud-computing "state-of-the-art" version. Section 5 concludes our paper. Acknowledgements and references can be found in Sections 6 and 7, respectively.

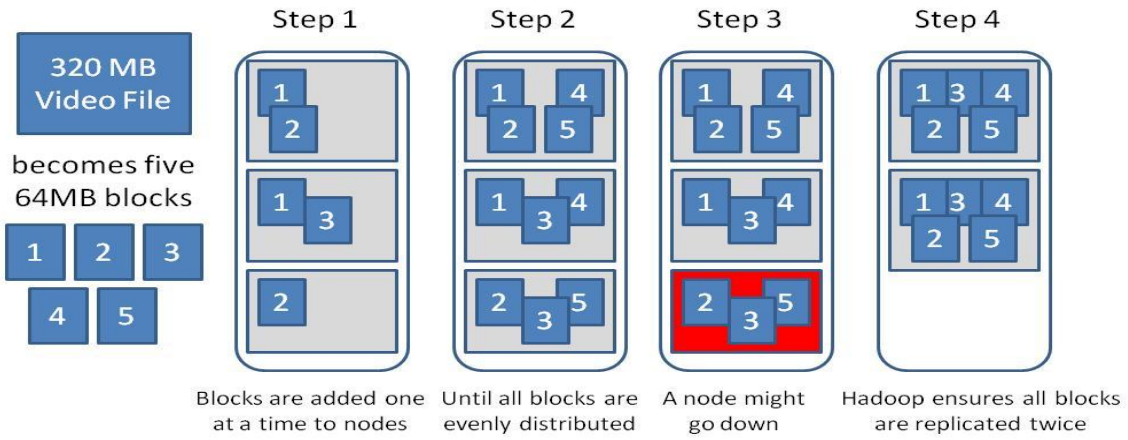
2.0 (U) Background: Bringing Vision to the Big-Data Cloud

(U) We review the state-of-the-art in big data cloud computing and computer vision. We conclude the section by answering the question, "How do we take computer vision to the cloud?"

2.1. (U) Apache Hadoop: Big-Data Cloud Computing, Map/Reduce and Distributed File Systems

(U) In [6], Google™ provided to the world the first glimpse at Map/Reduce, a batch processing programming paradigm that allows for big-data processing to be done on a distributed cluster of computational nodes (e.g. a cloud computing environment). Map/Reduce paired with a distributed file system overcomes the challenges described in section 1.0 of hard disk read-latency. It allows algorithms to run in parallel on different "blocks" (typically 64MB in size) of the data corpus spread across multiple hard drives in a data center. This is done through two phases: Mappers and Reducers. Mappers are first and they run their algorithm on the data block itself, read from a hard-disk on one of the nodes in the cloud computing environment. Mappers take in their data (usually pre-formatted) as `{Key, Value}` pairs and Values outputted from the Mapper and feeds them into the Reducer by sending it one key with a list of Values. Reducers then emit zero or more `{Key, Value}` pairs for every `{Key, List[Value]}` they take in.

(U) In 2008, Doug Cutting created what is today the most widely used free and open source version of Map/Reduce called Apache Hadoop [2]. Hadoop follows the paradigm set up by Google™ by providing a Java-based API to programmers to write Mappers and Reducers. Hadoop also comes with a distributed file system, the Hadoop Distributed File System (HDFS), which partitions data into the aforementioned block-sized chunks. In Figure 2, a step-by-step depiction shows how a 320MB WAMI video data file would be partitioned into five different data blocks of 64MB in size. As shown in steps 3 and 4, HDFS is resilient in that it automatically handles the loss of one data node by re-distributing data blocks when Hadoop detects that blocks are "under-replicated". When the node returns, it puts blocks back to what is depicted in step 2. This follows good-sense engineering because real life cloud computing environments are rarely ever at 100% node availability.



(Unclassified)

Figure 2. (U) A Depiction of Hadoop's Distributed File System Using a Sample 320 MB Video File

(U) Lockheed Martin ATL has experience in working with Hadoop starting in 2010. Since then, ATL used Hadoop for data processing and mining, social network graph construction, graph analytics and graph clustering. Consequently, we also have experience in related Hadoop-ecosystem technologies too: Accumulo [1], Zookeeper [5], Mahout [3], and NoSQL (e.g. Apache Pig [4]). When starting the HVISRE project, we needed a big-data cloud-computing environment for processing video data to implement an experiment for. Hadoop seemed like a logical choice based on experience, popularity and support, and a license that is easily applicable for our customer's needs. Over the years since, Hadoop has grown and become easier to use multiple languages, data formats, and scheduling customizations. It is a good choice to handle heterogeneous data such as collections of video and imagery of varying qualities, resolutions, formats and sizes.

2.2. (U) Computer Vision and Parallelism

(U) Computer vision is being used today to solve all sorts of interesting questions from object and activity recognition to image stitching to form a Wide Area Motion Intelligence (WAMI) picture.

(U) ATL has a broad range of experience in designing experimental computer vision systems that operate on a variety of platforms and modalities, developing content-based retrieval (CBVR/CBIR) algorithms and descriptor fusion techniques. We have an automated indexing and search capability for previously un-analyzed Predator video using video and image features. ATL similarly developed brain-inspired vision techniques for complex pattern recognition on LIDAR data. ATL researchers have worked (U) closely with a group at the University of Southern California on applying visual saliency techniques to datasets of analyst interest [11].

(

U) We classify Computer Vision today as a set of broad categories:

1. (U) Image-Stitching: Image stitching (or mosaicing) is a unique application for cloud deployment in that the majority of applications are only concerned with the boundary conditions on the edges of image frames. Assuming a rectangular frame, the brute-force approach attempts to find matches for each of the four tile boundaries by computing pixel-wise similarity along the boundary, up to a certain number of pixels deep; at best, this scales linearly with the number and dimensions of tiles [19].
2. (U) Content-based Image Retrieval: This class of algorithms generally attempts to perform matching between a user-provided sample input image (or video clip) and every other video in a database. In the past, we have addressed this by storing a set of compact, reusable descriptors computed during data ingest, and then using this compact representation for matching.
3. (U) Scene Understanding: Scene understanding (also known as scene classification) is a coarse classification of the contents of a scene using descriptors comprised of low-level features. Examples of this type of application include discriminating between types of natural scenery (i.e. beach, forest, desert, etc.) or man-made environments such as cityscapes. The Visual Saliency algorithm discussed in detail in this paper is in this class of algorithms. It should be noted that by "understanding" we refer not to a detailed dissection of the semantic contents of scenes but to a coarse high-level judgment that can be useful, for example, in cueing downstream algorithms for fine-grained analysis.
4. (U) Activity Recognition: This class of computer vision algorithms detects activities occurring inside of video streams by looking at multiple frames. This class of algorithm works on multiple levels. Starting with the bottom: *activities* are found and done so though examining immediate behaviors executed by a single entity. Then, at the next level *events* are found as collections of activities that describe a coherent, higher level action possibly spanning a larger time scale and multiple agents. At the top-most level, a *scenario* or *threat* is a collection of events that typically occurs over a time span greater than one day where events do not need to be spatially or temporally co-located.
5. (U) 3-D Reconstruction: 3-D reconstruction of 2-D images has many applications in spatial reasoning today. A certain amount of global knowledge of the scene is necessary to proper geo-registration of 3-D models; however many modern techniques use an over-segmentation method and concentrate on stitching together adjacent segments at a local level [19].
6. (U) Object Recognition: 2-D object recognition is the classic computer vision pattern recognition application. The traditional approach uses convolutional pattern matching to match a candidate pattern against every possible window on an image at multiple scales [20].

2.3. (U) What Makes Computer Vision Feasible in Apache Hadoop?

(U) When considering Map/Reduce, computer vision algorithms face a unique challenge for parallel implementation. From an algorithmic perspective, the fundamental problem is choosing the computational unit along which to divide processing; frames of a video, individual pixels of a frame, or even parameters across multiple iterations over the same pixel. To solve the big data problem, the algorithm designer would ideally divide computation amongst pixel-based subunits of the input capable of being hosted in a distributed file system such as Hadoop's Distributed File System (HDFS) [2]. This introduces the mathematical problem that pattern recognition algorithms generally have a need for normalized feature detection from across a single image frame. Frequently, in object detection and automated target recognition (ATR) tasks, the algorithm output is meant to be a list of *matches* to the candidate target along with their locations in the frame and, importantly, a measure of relative confidence that the target pattern has been detected. In general, a match for such an algorithm is computed with respect to global features; a match or positive detection is therefore a threshold on a normalized manifold (U) describing all pixels (and pixel neighborhoods) of the image with respect to the candidate pattern. If a single wide-area image is to be partitioned across independently executing ATR algorithms on different computational nodes, this relative confidence can no longer be computed. The best target "match" result on one partition of the frame may therefore be far inferior to the next best match in an adjacent partition [11, 16].

(U) The problem is complicated further where the input consists not only of a single wide-area image, but of a time sequence of images, i.e. a video. The usual problems with single-node detection and tracking of three dimensional target objects with occlusion and suboptimal pixels-on-target in video persist. The difficulty in computing normalized correlation is compounded along a third dimension, time, that for many classes of ATR and tracking algorithms may prove intractable. Video also contains a non-algorithmic challenge for distributed deployment in that the partitioning problem itself is far more complex; many video encoding schemes require that the entire video file be read in order to be decoded, which eliminates the possibility of a straightforward chunking of input files for random access. We discuss our solution to this when explaining our implementation in Section 4 [11].

(U) One key insight for distributed deployment of vision algorithms is that, though "computer vision" spans a tremendous range of operations with varying goals depending on the mission and input modality the vast majority of techniques which fall under the vision umbrella do have some separable computational components. For Map/Reduce deployment, we focus our research on those techniques, which from a feature extraction standpoint rely heavily on *local knowledge* features (i.e. a pixel and its immediate neighborhood) with static normalization needs, and the analogous case for spatiotemporal slices through video (voxels). The best candidates for implementation in a Map/Reduce paradigm are algorithms, which do the bulk of the feature extraction work locally without relying on correlations with features that may be extracted from adjacent HDFS data blocks.

(U) The task of determining which computer vision algorithms are more amenable to distributed implementation generally and specifically suitable for implementation in the Hadoop Map/Reduce framework largely focuses on which components of the algorithm may be cleanly executed on individual frames (of a video) or regions (of a frame), on distributed nodes, without affecting the overall integrity of the result. Algorithms that contain a deep stack of feature extraction that is not integrated as global features until the end of the pipeline may, as a rule, be more easily adapted to distributed implementations. The dichotomy includes algorithms which depend on local vs. global knowledge, those which contain inherently or "embarrassingly" parallel operations, and in the case of video, those which rely on features computed from individual frames vs. spatio-temporal voxels that may cut across naively separated regions. The following sub-sections describe the metrics and their algorithm classes.

2.3.1 (U) Local vs. Global Knowledge Features

(U) Local features may be computed using only a portion of the data stream independent of other instances. These algorithm classes need little or no sharing of intermediate data products between multiple instances to derive an intended conclusion. On the opposite side of the spectrum, algorithm classes that require a lot of shared mutual knowledge between multiple instances are considered to require global knowledge. It is relatively easy to see why local knowledge algorithms fare better than global knowledge ones in a distributed paradigm like Map/Reduce. Local knowledge based algorithms typically take less advantage of Map/Reduce'sSort and Shuffle Phase between Mappers and Reducers and thus require fewer Reducers. Local knowledge algorithms require the least thought to how they need to cooperate. In contrast, algorithms which depend on global knowledge are much more difficult to parallelize. These algorithms often times need to share a lot of intermediate knowledge between instances. In this metric, Content-based Image Retrieval and Image-Stitching algorithms are the most feasible, i.e. [21].

2.3.2 (U) Inherent Parallelism

(U) Many algorithm classes already exhibit parallel properties while others do not. We classify algorithms in one of three areas: multi-process, multi-threaded, and single-threaded. Algorithms that are already multi-process typically also are multi-threaded. These algorithms are the easiest to implement in Map/Reduce largely because they have already solved global knowledge problems through various means (writing to disk, sockets to a central server, etc.) Aside from multi-process, Multi-threaded algorithms are the simplest to implement. These algorithms typically share information between one another through shared memory. This paradigm is straightforward enough to break into Map/Reduce because of the Key-Sort phase between Mappers and Reducers mentioned in Section 2.1. Within this phase, memory that is stored between instances of the algorithm is pushed on to the network bus. Then, a portion of the algorithm is implemented in the Reducer, which interprets and processes data from the multiple input Mappers. Finally, single-threaded algorithms are the most difficult to break apart. A solution for how to partition memory or shared knowledge typically has to be invented, sometimes making the specific algorithm within the class nearly impossible to implement in Map/Reduce. In this metric, Scene Understanding and Image-Stitching algorithms are typically the most feasible, i.e. [9, 11, 12].

2.3.3 (U) Dynamic vs. Static Normalization

(U) Dynamic and static normalization refer to the peculiarity of many computer vision algorithms that a local feature vector computed in a single part of an image must generally be normalized in order to be used for mapping against other descriptors and feature vectors; more significantly, the process of computing a "match" for a template pattern in an image will generally lead to scores which are not comparable across runs. In this metric, Content-based Image Retrieval and Image-Stitching algorithms are the most feasible.

2.3.4 (U) Spatio-Temporal Voxels

(U) Certain algorithms require that they process multiple frames (stacks) in order to make a conclusion. These algorithms use spatio-temporal descriptors. Other algorithms only operate on a single frame at a time and have purely spatial descriptors. Descriptors are amalgamated instances of feature representations, where a single descriptor may be a representation consisting of several computed features. For certain problems in video, particularly activity detection and matching, it makes sense to compute descriptors which describe the contents of not just a single frame but of a spatiotemporal "slice" of video consisting of a temporally contiguous subsequence of frames. The challenge of computing such features in Map/Reduce would primarily involve designing an efficient representation of data storage to allow the data from one or a subset of frames to be efficiently buffered and read in serial. Once done, voxel-based algorithms are more feasible to implement in the big-data cloud-computing environment. For this metric, only Activity Recognition algorithms have single-frame descriptors and only some of them could be put through this transformation process, i.e. [10].

3.0 (U) Visual Saliency and Hadoop for Wide Area Motion Intelligence

(U) One problem that the WAMI community is seeking solutions to is for a new way to sift through millions of images and detect patterns that are interesting. Imagine that a single very high quality camera is taking a giga-pixel video at 8hz from an aerial platform. The data output of this camera is huge, but only a fraction of it is actually useful information for an analyst to build knowledge off of. If the analyst could identify features indicative of tactically relevant portions of the incoming imagery they could increase their performance and accuracy. It is in this space that a Hadoop version of the pattern-recognition Visual Saliency algorithm is needed [9, 11].

(U) Visual saliency is based on the neuromimetic attention of mammalian visual cortexes [9, 11, 16]. It operates in the same way that the human eye recognizes feature patterns of objects and physical space boundaries (e.g. a room). Saliency extracts those features and helps determine what might actually be

(U) interesting to a human in detecting what is called *saliency features.* This is done through classifying each frame of video as follows:

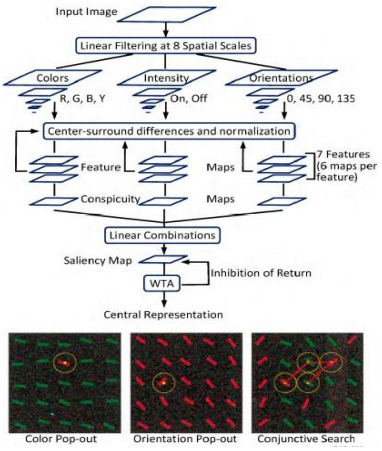
- (U) Color: Color disparity maps (2-dimensional arrays) are computed from an image. Wherever intensities of these primaries exist in the absence of one or more of the others, a salient feature is found. This type of classifier is best when detecting objects that have few colors that are unlike other objects near them.
- (U) Intensity: Brightness (luminance disparity) maps of a frame are computed. Through comparison of neighboring pixels, salient features can be detected. Often times, this classifier is the one most heavily used when determining if an object is man-made or not because it easily picks up reflective surfaces found in WAMI video.
- (U) Orientation: This classifier attempts to look for similar pixels in neighboring areas of the frame. It determines if certain lines can be formed and with what orientation (0 to 359 degrees) the lines have. This classifier is best suited to help determine where objects or surfaces end compared to other objects or surfaces in a frame.

(U) Visual saliency is a good candidate for conversion to Map/Reduce because it processes data by looking at the neighborhood of a pixel and thus does not directly rely upon global image features (see Section 2.3.1). Our implementation of the algorithm can also compute these salient feature types in parallel, taking advantage of multiple cores of a processor.

3.1. (U) Saliency in Map/Reduce

(U) Based on an existing implementation of the Visual Saliency algorithm, we designed and implemented a version of it in Hadoop using Map/Reduce and the HDFS as described in Section 2.1. We use Map/Reduce in three phases to implement saliency: the ingest exchange-transfer-load (ETL) phase, the analytic phase, and the meta-data tagging phase.

(U) In the ETL Phase, we use Map/Reduce to build saliency maps of an entire data corpus of imagery that is being ingested into the cloud for the first time. Saliency Maps are best described as heat-maps that the most salient images can be found. Our implementation of the ETL algorithm is depicted in Figure 3. Our ETL Mapper depicts color, intensity and orientation of single frames of videos. The reducer collects these and combines them into a Saliency Map. After a single pass through of the algorithm, our implementation can build coherent Saliency Maps of an entire video corpus based on examining each frame's pixels and their neighbors. This phase only needs to be done once, as it is the most costly. Our visual saliency implementation is non-deterministic, meaning that Saliency Maps generated from images in this phase do not need to be regenerated ever. After the ETL finishes, we have a sense as to which image frames in a data corpus contain the most visually salient objects that would draw the attention of a human eye.



(Unclassified)

Figure 3. (U) An Overview of the ETL Saliency Map/Reduce Algorithm

(U) The Analytic phase is done after the ETL completes. At this point, the saliency maps are now stored back onto HDFS and used for analysis in subsequent Map/Reduce jobs. Analytics are usually done on behalf of user-based requests. There are many different kinds of analytic questions that one could pose on a set of saliency maps. For example, "What frames have the most salient features that are man-made?" is a good question to ask of a video data set that is shot largely in the desert where few man-made objects exist. Visual saliency can answer a question like that one because man-made objects are usually reflective (e.g. have high-intensity)

(U) and are distinct in color in comparison to near-by pixels. The Map/Reduce jobs here can be as simple as comparing two saliency maps together to complex multi-step jobs that cluster like saliency maps together. We discuss an implementation of one analytic Map/Reduce job we created later in Section 4.1.

(U) In the Meta-data Phase we solve an additional problem by assigning meta-data to tag videos whenever questions are asked in the analytic algorithms. This is very important because once meta-data is assigned to a video or its frames, it is much quicker to retrieve for user consumption without having to launch a time consuming analytic Map/Reduce job. The results of these Map/Reduce jobs are descriptors usually posted into a quick-access and indexed relational database since they do not take up a lot of space. These databases can then queried by a multitude of analytic tools such as Analyst's Notebook [8] since these tools often rely upon key-word search.

3.2. (U) A Walkthrough: Finding Man-Made Objects in Large WAMI Videos

(U) To better describe the process, we provide a walk-through of our implementation. Imagine that a UAV has produced a lot of video imagery and we want to perform analytics on it. The first ETL phase takes the video and builds saliency maps out of it as described above. This data shows that in each frame of the video which images are most salient (e.g. the most interesting to look at that a human eye might be drawn to). After this phase, we know immediately which frames stand out more so than others so we can then move on to the analytics phase.

(U) Now the analyst may want to be interested in salient images that have the characteristics of man-made objects such as a vehicle. For example, a white truck is a good salient object as it has distinct color, orientation, and intensity features as shown in Figure 4. When this type of image is first ingested into the data corpus, it too has a saliency map generated using the same ETL process. Analysts insert information about that truck using meta-data indicating that it is a man-made object. With that in mind, the next question that an analyst can ask is, "what frames of the new video contain the most man-made objects" using the truck as an exemplar.

(U) Consider a frame of the UAV video in Figure 5, a WAMI frame at NIIRS level 8+ where one frame is larger than 32MB. In this case, a Map/Reduce job is launched and the saliency image of the truck is compared to one generated from this sample video frame. The Mapper here does the comparison between saliency maps and then emits probabilities of likeness. The Reducer collects the probabilities and sorts them, only returning the highest probabilities. The analyst's question is resolved because we can tell what frames happen to have something in them that most resemble the salient features of a white truck.

(Unclassified)

Figure 4. (U) A Sample White Truck and its Salient Features

(Unclassified)

Figure 5. (U) A Sample WAMI Image of a Town

(U) Finally in the meta-data phase, we take the experience of the analyst and provide the same meta-data descriptor they made of the truck to the particular frames where they were found. If provided, we can

(U) use existing meta-data from the UAV platform to tie in a time and geo-spatial information too. This is done through using a Mapper that looks at a frame's global ID and finds it in the data corpus. The Reducer writes to a relational database where salient features like a man-made truck can be found. In the future, this meta-data descriptor catalogue can be queried without having to look at the entire data-corpus again.

3.3. (U) Implementation Challenges in Hadoop

(U) As with any cloud-computing implementation, challenges arise and we would be remiss if we did not write about them and how we addressed them. First off, most videos are compressed and require that they are decompressed and stored effectively in HDFS chunks (see Section 2.1). Our initial solution is to store the data raw. However, others have been addressing this problem in [12]. Another problem is how we overcome splitting videos effectively between data chunks on HDFS. In our approach, because we have the flexibility of a custom raw-storage we use Hadoop Map Files, which are similar to hashmap data structures. Within the Map File, we store global-frame IDs as keys while keeping the values as the image's raw bytes. A final challenge is that like most computer vision algorithms, visual saliency requires the use of C libraries while Hadoop is purely Java-based. We use Apache Avro to represent data in a language independent fashion and then rely upon a custom implementation of Hadoop Streaming to feed data frames into our C++ version of the ETL saliency-map generation algorithm. Analytic and meta-data Map/Reduce jobs are typically written purely in Java, requiring no language barrier to be crossed.

4.0 (U) An Experiment: Finding Interesting Features in Sparse GISR Video Data

(U) We took our implementation of the Visual Saliency algorithm and applied it to an experiment using real data. We set out to prove that a Map/Reduce version of the Visual Saliency algorithm is no different than just running the current "state-of-the-art" version of the algorithm with the same data. We also set out to prove that Map/Reduce versions of the algorithm perform better on larger data files, as we should expect of a Hadoop big-data cloud.

4.1. (U) Experiment Data and Implementation

(U) The data we selected for our experiment comes is GISR data with permission given to us to use (credited in Section 6). Even though our data set has WAMI video in it, it is not at the level of NIIRS 6+, which we need to identify salient features of white-truck sized objects. Thus, we chose narrow-field of view images for now. However, we fully expect in the future that WAMI video will become high quality enough to see salient features such as trucks. At that time, the exact same experiment described here can be run again and the same algorithms applied.

(U) From the GISR video data, we made two video data corpuses:

- (U) A set of 15 Blue-Devil videos. Each video is approximately 60 seconds long and ranges from 20 to 45 MB in size, compressed. The videos shot over a desert terrain containing only a small amount of man-made objects. When stored in uncompressed format on HDFS, this corpus is about 3GB in size in full.
- (U) A set of five Blue-Devil videos, five Penguin UAV videos [15], and five FLIR Thermal videos [7]. The Penguin UAV videos are approximately 60 seconds long and range from 18 to 36 MB in size, compressed. The FLIR Thermal videos are approximately 60 seconds long and range from 40 to 46 MB size, compressed. These videos are also shot over a desert terrain containing only a small amount of man-made objects. When stored in uncompressed format on HDFS, this corpus is about 5GB in size in full.

(U) We implemented the ETL Map/Reduce algorithm and one analytic Map/Reduce program as described in Section 3.1. The ETL algorithm scans through the video data corpus and computes saliency maps describing what might actually be interesting in every video. Since the video is shot largely in the desert, contrasting features always stand out in the saliency maps computed during the ETL phase of our experiment.

(U) The analytic Map/Reduce algorithm answers the question, "Which Saliency Map in the entire corpus is most similar to a target image?" This algorithm requires an input "target image," which it creates a saliency map out of to compare all other saliency maps in the corpus to. Usually an interesting frame containing man-made objects is supplied for this comparison. In our case, we found an image in a Blue-Devil video of a white truck in the middle of the desert, which we set aside as the target image for our experiment.

(U) We also implemented a "baseline" version of the Visual Saliency algorithm that does not use Map/Reduce to test our hypothesis described in section 4.2 below. It is designed to run on a single node and to process video files in a data corpus in sequential order. This baseline algorithm does the work of both the ETL and analytic Map/Reduce algorithms described above.

(U) Our implementation did not include a Map/Reduce version of a Meta-Data tagging algorithm; however we are using an extension of the Metadata Extraction and Tagging Service Ontology (METS) [13] to associate the target image with any closely-identified frame in the video corpus after processing finishes.

(U) The cloud-computing environment we used has four data and task nodes running Fedora Core 12 and one node dedicated to Hadoop's namenode, job-tracker, and the secondary namenode. The nodes each have 8GB RAM with Intel Xeon CPUs with dual-core processors at 2.66GHz. Hadoop was configured with a capacity of 15GB of HDFS space. Hadoop's job tracker which runs Map/Reduce job was left in the default configuration of running a maximum of two mappers and two reducers on one node simultaneously.

4.2. (U) Experiment Metrics and Results

(U) In our experiment, we attempt to answer two questions through four experiments:

- (U) Can a Map/Reduce version of the Visual Saliency algorithm perform just as well as a baseline version of the algorithm that does not take advantage of the cloud computing resources? This question is the focus of experiments A and B.
- (U) Does the Map/Reduce version of Visual Saliency process video imagery faster? We certainly would expect that a cloud-based version of the algorithm would process more data quickly because it takes advantage of reading data off of multiple disks simultaneously. This question is the focus of experiments C, D, and E.

(U) **Experiment A.** In the first experiment run, we ran our baseline algorithm on the 15 Blue-Devil video data set as described in Section 4.1. The results are in Table 2. In all cases, the baseline algorithm picked out that frame #720 of video #10 as the highest probability of detection and that the frames #712 and #728 of video #10 as the next highest. When we ran the Map/Reduce version, the results were identical.

Table 2. (U) Results of Experiment A on the 15 Blue-Devil Video Set

(Unclassified)	Highest Match	Second Highest	Third Highest
Baseline Saliency Algorithm, when run on the 15 Blue-Devil video set.	Video-10, Frame 720	Video-10, Frame 712	Video-10, Frame 728
Map/Reduce Saliency Algorithm, when run on the 15 Blue-Devil video set.	Video-10, Frame 720	Video-10, Frame 712	Video-10, Frame 728

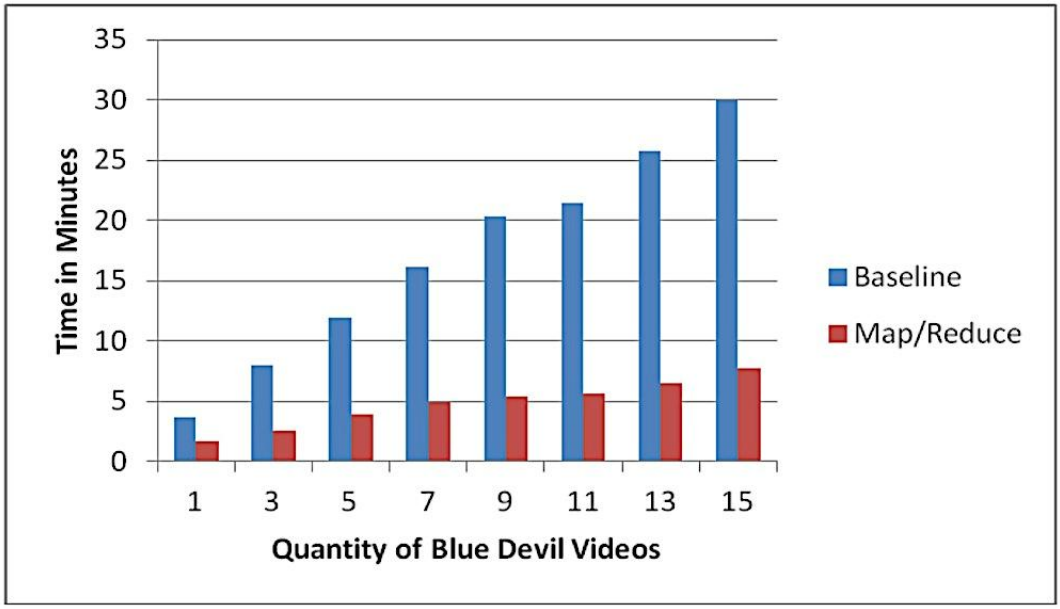
(U) **Experiment B.** In this experiment, we used the mixed data set described in Section 4.1 to show that the algorithms can process heterogeneous videos of mixed qualities and sizes. As a reminder, the mixed data set has a third of its videos shot from the Blue-Devil and two thirds of the videos shot from another platform. See Table 3 for the results. Both the baseline and the Map/Reduce versions are identical. The Blue-Devil Video-5 is the same as the Video-10 above.

Table 3. (U) Results of Experiment B on the mixed video set.

	Highest Match	Second Highest	Third Highest
Baseline Saliency Algorithm, when run on the mixed video set.	Blue-Devil-5, Frame 720	Blue-Devil-5, Frame 712	Blue-Devil-5, Frame 728
Map/Reduce Saliency Algorithm, when run on the mixed video set.	Blue-Devil-5, Frame 720	Blue-Devil-5, Frame 712	Blue-Devil-5, Frame 728

(Unclassified)

(U) **Experiment C.** To answer the second question, "Does the Map/Reduce version of Visual Saliency process video imagery faster?" we measured the start and stop times for both the baseline and the Map/Reduce version of Visual Saliency and took the difference. In the first set of runs, we timed the baseline algorithm on the 15 Blue-Devil video data set in increments of one, three, five, seven, nine, eleven, thirteen, and the full fifteen videos. In the second set of runs, we timed the Map/Reduce version in the same set of increments of the 15 Blue-Devil video data set. The results are shown in Figure 6.

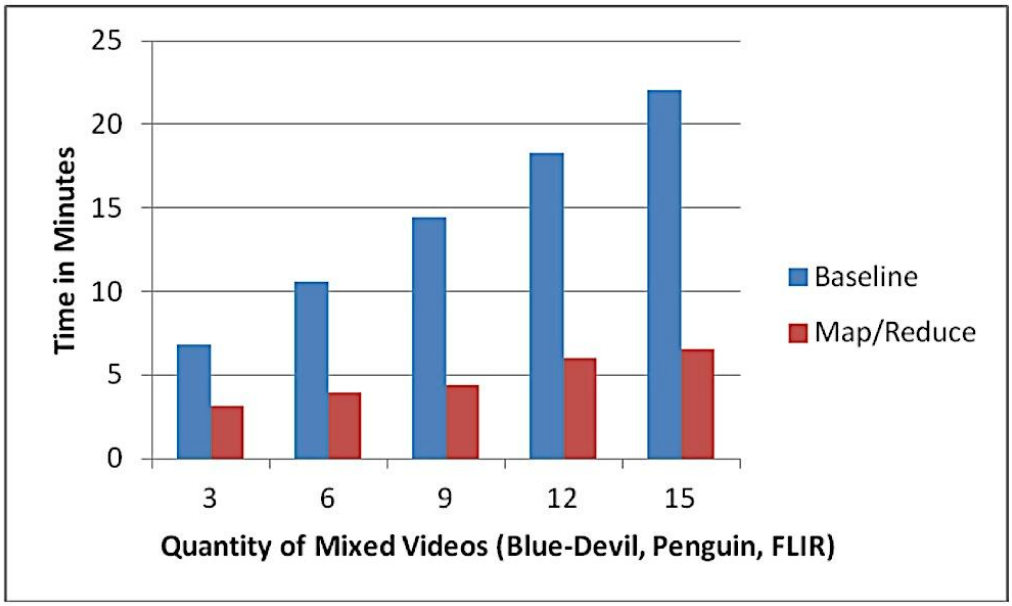


(Unclassified)

Figure 6. (U) Performance of Baseline vs. Map/Reduce on Blue-Devil Data Set

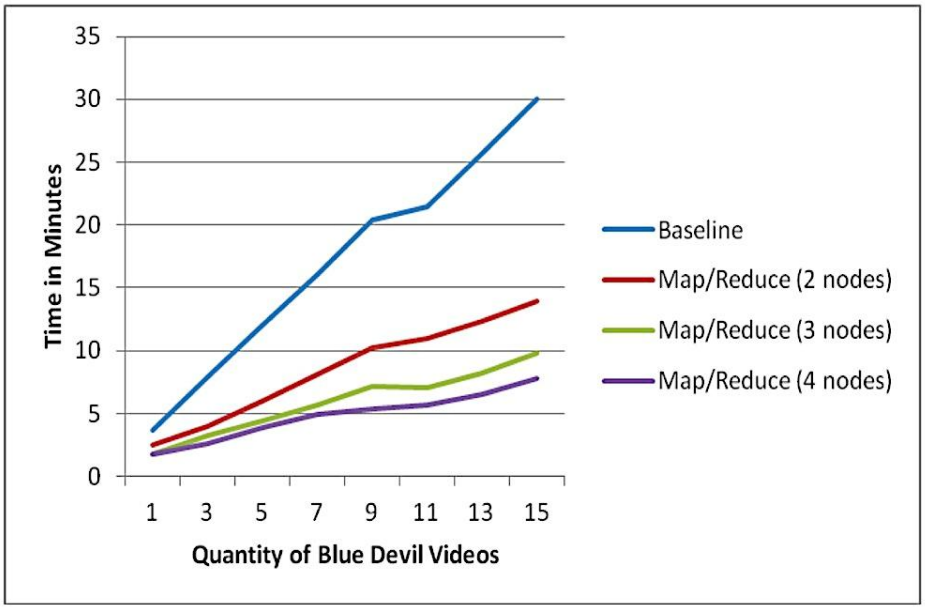
(U) **Experiment D.** We asked the same question in experiment D but on the mixed-data set described in Section 4.1. In this case, we ran it on a mix of three, six, nine, twelve and the full fifteen mixed videos. The results are shown in Figure 7.

(U) **Experiment E.** In this experiment, we showed how much of an increase of performance the Map/Reduce version of the algorithm has when reducing the quantity of taskable nodes from the full four to three, and two. We re-ran Experiment A on the Blue-Devil data set to show exactly how much performance is gained by every node added into the experiment. The results are shown in Figure 8.



(Unclassified)

Figure 7. (U) Performance of Baseline vs. Map/Reduce on the Mixed Video Data Set



(Unclassified)

Figure 8. (U) Comparing Two, Three, and Four Nodes of Map/Reduce to the Baseline

4.3. (U) Analysis

(U) In Section 4.2, experiments A and B tell us that the Map/Reduce version of Visual Saliency does no harm when comparing its quality of result to the baseline "state-of-the-art" version. Although this validation step is useful, it proves an important point: Scene Understanding algorithms such as Visual Saliency can indeed be run in a big-data cloud using Map/Reduce.

(U) Experiments C and D show that unequivocally the Map/Reduce version of Visual Saliency runs more quickly than a single-node version as shown in Figures 6 and 7. This shows that the Hadoop

(U) infrastructure, when properly given enough nodes, can run CV algorithms faster than the state-of-the art. Even though the data corpus of experiment D is larger than that of C, both the baseline and Map/Reduce versions run quicker. This is because in both cases, we automatically throw out comparing video frames to one another if the size and content of their saliency maps is significantly different.

(U) The differences between the baseline and Map/Reduce version become more significant as the size of the data corpus increases. In Figure 8, Experiment E shows that when comparing Map/Reduce at to the baseline version even adding only one additional node of processing can make a large impact on processing time. It also shows that Map/Reduce is not without overhead cost. In the Map/Reduce runs, each node should be processing two mappers or two reducers simultaneously where the baseline algorithm only uses a single processor. This means we would expect that the Map/Reduce job to run four times as fast on the 2-node experiment, six times as fast on the 3-node experiment and eight times as fast on the 4-node experiment. However, as Figure 8 shows, the overhead of Map/Reduce is diminishing as we add more videos to the data corpus and eventually it will be eclipsed by the sheer amount of time a baseline algorithm would take to run.

5.0 (U) Conclusion

(U) There are simply too many sensors producing too much video data to be analyzed in a reasonable amount of time. Low-level CV processing capabilities can sift through large quantities of video data to highlight the interesting pieces. However, this can only be useful if it can be done quickly through an approach that scales to meet the increasingly large sizes of video corpuses. Such an approach must overcome the difficulties of hard disk read speeds and capacities while also remaining financially viable for the long term. In this paper, we've shown and validated an approach where this is possible using a big-data cloud-computing environment; if only for a minimum of one algorithm: Visual Saliency. We have also shown that there are a lot of reasons to believe that other types of CV algorithms are also feasible to be brought in to the same environment. Yet, a significant number of problems in both research and engineering nature exist such as effective data storage and indexing of video on distributed file systems and integrating with algorithms written in multiple languages, perhaps taking advantage of GPUs. Certainly, our work calls out for a new advent of ideas in the CV community on how to move beyond the world of small quantities of video and enter the real ISR world of big and heterogeneous video corpuses.

6.0 (U) Acknowledgements

(U) We would like to acknowledge and thank Benjamin Cutler (benjamin.cutler@darpa.mil) for his permission to use GISR video data for purposes of experimental testing of our algorithms.

7.0 (U) References

[1] Apache Acumulo (2011) http://accumulo.apache.org

[2] Apache Hadoop (2008-2012) http://hadoop.apache.org

[3] Apache Mahout. (2011) http://mahout.apache.org

[4] Apache Pig (2007-2012) http://pig.apache.org

[5] Apache Zookeeper (2010) http://zookeeper.apache.org

[6] Dean, J. and Ghemawat S. "MapReduce: Simplified Data Processing on Large Clusters" (2004) In Proceedings of the Sixth Symposium on Operating System Design and Implementation.

[7] FLIR Thermal Imaging: http://www.flir.com

[8] IBM i2 Analyst's Notebook. http://www.i2group.com/us/products/analysis-product-line/ibm-i2-analysts-notebook.

[9] Itti L., "Models of Bottom-Up and Top-Down Visual Attention" California Institute of Technology, Jan 2000. [Ph.D. Thesis]

[10] Itti L., Baldi, P. F. "A Principled Approach to Detecting Surprising Events in Video" In the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 631-637, Jun 2005.

[11] Li Z., Itti L., "Saliency and Gist Features for Target Detection in Satellite Images" In the IEEE Transactions on Image Processing, Vol. 20, No. 7, pp. 2017-2029, 2011.

[12] Lin, Y. et al. "Large-scale Image Classification: Fast Feature Extraction and SVM Training" (2011) In Proceedings of the Conference on Computer Vision and Pattern Recognition. Pages 1689-1696.

[13] The Metadata Extraction and Tagging Service (METS) (2012), http://mets.egovservices.net/METS-ontologies.shtml

[14] National Image Interpretability Rating Scale (1998), http://www.fas.org/irp/imint/niirs.htm

[15] The Penguin UAV: http://www.uavfactory.com

[16] Pereria, et al. "An Architecture for Distributed High Performance Video Processing in the Cloud" (2010) In Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing.

[17] Oracle's Exadata X2-8 HP Full Rack (September 2012). http://www.oracle.com/technetwork/server-storage/engineered-systems/exadata/dbmachine-x2-8-datasheet-173705.pdf and http://www.oracle.com/us/corporate/pricing/exadata-pricelist-070598.pdf

[18] Siagian C., et al. "Beobot 2.0: Cluster Architecture for Mobile Robotics" In the Journal of Field Robotics, Vol. 28, No. 2, pp. 278-302, March/April 2011.

[19] Sexana, A. et al. "Learning 3-D Scene Structure from a Single Still Image" (2007) In Proceedings of ICCV workshop on 3D Representation for Recognition (3dRR-07).

[20] Sun, Z. et al. "Object Detection using Feature Subset Selection" (2004). In the Journal of the Pattern Recognition Society. Published by Elsevier Ltd.

[21] Szeliski, R. "Image Alignment and Stitching" In Paragios, N. et al., editors, Handbook of Mathematical Models in Computer Vision. Pages 273–292. Pub.lished by Springer (2005). ISBN 0387263713

[22] Tang K., et al. "Learning Latent Temporal Structure for Complex Event Detection" (2012) In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[23] White, T. "Hadoop: The Definitive Guide" 3rd Edition. (May 2012) O'Reilly Media and Yahoo Press. ISBN: 13: 978-1449311520.